Accessibility Study of Rich Web Interface Components

Juliana Cristina Braga
Universidade Federal do ABC
Santo André, Brazil
e-mail: juliana.braga@ufabc.edu.br

Rodrigo Torres Leme
Universidade Federal do ABC
Santo André, Brazil
e-mail: rodrigo.torresleme@gmail.com

Rafael Jeferson Pezzuto Damaceno
Universidade Federal do ABC
Santo André, Brazil
e-mail: rafael.damaceno@aluno.ufabc.edu.br

Silvia Dotta
Universidade Federal do ABC
Santo André, Brazil
e-mail: silvia.dotta@ufabc.edu.br

Abstract – **The use of the latest technologies to develop rich interfaces for websites could decrease your accessibility, making problematic the access for people with disabilities. Faced with this problem, this paper executed a study to evaluate the accessibility of rich components on the Web. The paper used WAI-ARIA recommendations as reference and analyzed the accessibility problems found in the Jboss Richfaces components library. The study demonstrates a methodology to make the analyzed rich components accessible.**

Keywords – Rich Internet; accessibility; WAI-ARIA; Web 2.0.

I. INTRODUCTION

Accessibility is defined as the possibility and condition range, perception, and understanding for safely and autonomously use of buildings, space, furniture, urban equipment and elements [1]. The term indicates the possibility of any person to benefit from a life in society, including the internet.

Accessibility on the internet or accessibility on the web means to allow web access for everyone, not depending on the user type, situation or tool. It is concerned with creating or rendering tools and web pages accessible to a larger number of users, including people with disabilities.

The web is a resource that has become more important in many life aspects: education, work, government, commerce, health, leisure, and many more. It is essential that the web is accessible in order to allow people with disabilities to have equal access and opportunities. Also, an accessible web may help people with disabilities to be active part of the society.

However, with the changes in the sites development technologies, accessibility on the web has decreased, and could contribute to increase digital exclusion and to negatively impact the life of the disabled who already uses the internet.

Rich Internet Applications (RIAs) are web applications that employ advanced technologies of user interface to try to bring the interface of web systems closer to the interface of *Desktop* systems [2].

The earliest web applications supported basic HTML only, and although they could provide simple functions, these applications did not have nor provided the experience of a *Desktop* application.

The general objective of this research is to study solutions which may reduce the negative impacts that new rich technologies induce on the accessibility on the Web. Specifically, the objective of this paper is to study the accessibility of some components of the *Jboss Richface*s toolkit.

This paper is organized as follows. Section 2 introduces fundamental concepts to understand accessibility evaluation process of a website using rich components. Section 3 shows all technical items and steps that led this study. Sections 4 to 6 describe encountered problems and their resolution, added to the system validation. Section 7 presents the conclusions of this work.

II. THEORETICAL BACKGROUND

A. RIAs Technologies

There are two technologies that facilitate the RIAs development: *toolkits* and *frameworks. Toolkits* assist on abstracting the differences of browsers and on providing basic functions to handle events. *Frameworks* assist on standardizing sites development. Some *toolkits* and *frameworks* examples are: Dojo, Web Toolkit Google, Dojo Abra Rico, Prototype, User Interface Library, Zimbra Kabuki AJAX Toolkit, Yahoo User Interface, Jboss Richfaces, JavaFX, Ruby on Rails, and Java Server Faces (JSF).

This paper studies the *Jboss Richfaces 3.3.3.Final toolkit* and the *Java Server Faces 1.2 framework*, based on the World Wide Web Consortium (W3C) recommendations [3].

B. World Wide Web Consortium (W3C)

The *World Wide Web Consortium* (W3C) is an international consortium created in 1994, and it is responsible for establishing web standards. More than 400 organizations are members of this Consortium.

The W3C dealt with the accessibility problem in 1998 when the W3C Web Accessibility Initiative (WAI) was launched. Therefore, WAI is the part of W3C responsible for web accessibility.

WAI works with organizations from all over the world to develop strategies, guidelines and resources aiming to make the Web accessible to people with disabilities. One of the WAI roles is to develop guidelines and techniques that describe accessibility solutions for Web software and developers.

Until 2008 WAI did not handle accessibility problems in RIAs and from that year on it started to handle these problems by releasing the *Accessible Rich Internet Applications Suite* (WAI-ARIA 1.0).

C. WAI-ARIA

WAI-ARIA 1.0 is a set of technical recommendations that is being developed to specify ways to make the components of the rich internet accessible.

In fact, these recommendations suggest the insertion, in the sites source code, of tags of four types: *landmark*, *role*, *state* and *properties* [3].

Landmark is an element that indicates the *layout* divisions of a site, and can be header type, navigation bar, footer, and text body, among others [3].

Role designates the role executed by a RIA component [3]. As an example, a *slider* component, such as the one presented in Fig. 1, can be created from images and source code written in *JavaScript* language and can include the WAI-ARIA tags to make it accessible for a screen reader.

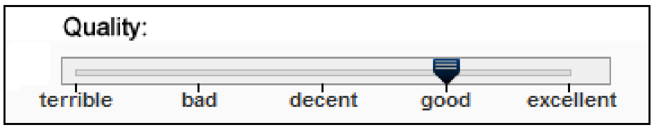

Fig. 1. Slider component created from images and JavaScript source code [3].

Properties indicate the property that a given component has [3]. In the example from Fig. 3, the “maximum-value” property could exist. *State* indicates the state of this property; in this example, the “maximum-value” *property* could have the “good” *state*.

A set of WAI-ARIA tags of *live* type also exists, and they are important to inform the screen reader about the updates without full page changes, which are common in sites that use RIAs [3]. An example is the decreasing count of remaining characters to be typed in a text box, a component that is very abundant in social networking services sites. Inserting the tag in this case would allow the screen reader to warn the user about how many characters are left to be typed.

D. Tools to Evaluate Accessibility

The evaluation of a given site accessibility can be done manually and/or automatically.

Manual evaluation consists of inspecting the site pages source code, in order to analyze its contents based on the W3C guidelines, searching for accessibility errors.

Automatic evaluation consists of testing the pages of a site using evaluation software that detects the code and analyzes their contents. Usually, the software is based on W3C guidelines, and evaluates the accessibility level inside these set of rules to produce detailed reports automatically. Generally, these tools/reports present the errors and suggestions on how to fix them, as well as the verifications that must be executed manually.

There are dozens of evaluators/validators of websites accessibility available in the market, such as: Cynthia Says [4], Hera [5], Ocawa [6], and W3C Validator [7].

E. Assistive Technologies

Inside Computer Science scope, assistive technologies are a set of software and hardware designed specifically to help people with disabilities to perform their daily activities [8].

That research studied voice synthesis assistive technology, also known as screen reader. This kind of computer program can reproduce audio of user interactions with the operational system and with others programs, like web browsers. Currently, there are many screen readers, such as: NonVisual Desktop Access (NVDA) [9], Jaws for Windows Screen Reading [10], Virtual Vision [11] and Window-Eyes [12].

III. STUDY ORGANIZATION

This project was developed in four steps, with some of them being concurrent. The first step was the survey and the bibliographic study already mentioned in the theoretical reference. The second step was the analysis of rich components accessibility; the third step was the adjustment of the rich components accessibility; and the fourth step was the validation of the study components accessibility. The following topics provide details of steps two to four.

IV. IDENTIFICATION OF THE ACCESSIBILITY PROBLEMS OF RICH COMPONENTS

This study analyses an evaluation software used in a university to measure teachers and disciplines performances. It uses the following rich components: *commandButton*, *commandLink*, *status*, *ajaxValidator*, *column*, *dataTable*, *message*, *messages*, *modalPanel* and *panel*.

The system consists of four pages: access page, with login and password; list of disciplines page, grading page and conclusion page. The source code of each page that uses the components is composed by four parts: HTML, Java Server Pages, Asynchronous JavaScript and XML (AJAX) and JBoss RichFaces.

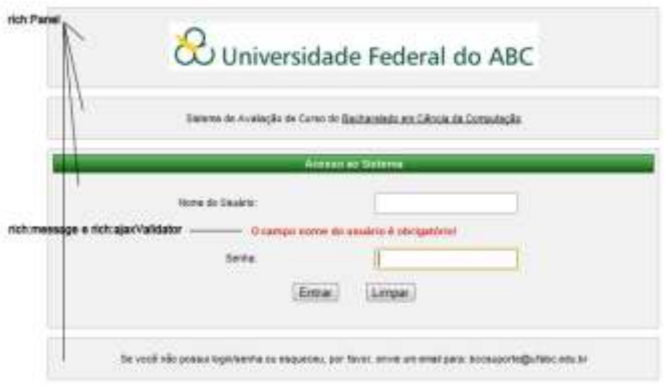
rich:Panel
Universidade Federal do ABC
Acesso ao Sistema
Nome do Usuário:
rich:message e rich:ajaxValidator
O campo nome do usuário é obrigatório!
Senha:
Entrar
Limpar

Fig. 2. The use of *JBoss RichFaces* components in the evaluation system under study. Access page. 2011.

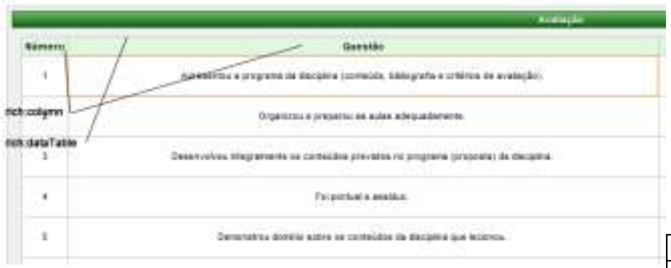

Fig. 3. Use of *JBoss RichFaces* components in the evaluation system under study. Evaluation page. 2011.

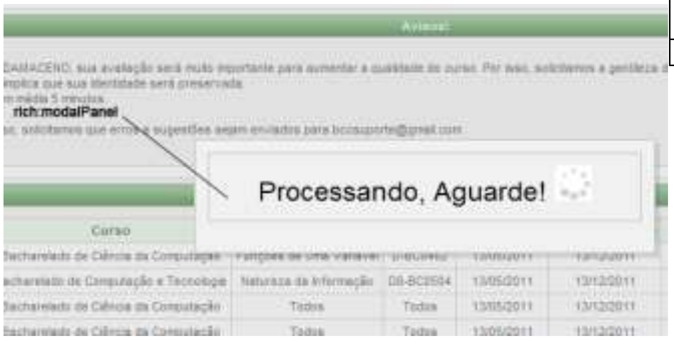

Fig. 4. Use of *JBoss RichFaces* components in the evaluation system under study. List of disciplines page. 2011.

It is known that an accessible site must firstly be in accordance with the HTML standards suggested by W3C. Therefore, in order to help the identification of the accessibility problems of the study components, an automatic tool for validation of sites suitability to W3C standards was used. The chosen tool was *Validator* from W3C [7], which is free software and does not require installation. This tool was selected due to being the only one in the category that supports WAI-ARIA tags, which are included in the Document Type Definition (DTD) file that defines the set of HTML tags the browser should recognize and interpret. It should be noted that there are other tools available for validation, such as the "Avaliador DaSilva" (Evaluator DaSilva) [13], and "Avaliador e Simulador de Acessibilidade de Sites" – ASES (Evaluator and Simulator of Accessibility of Sites) [14], which do not include (until now) the tags related to WAI-ARIA, making impossible any analysis.

For validation, the *"Validate by direct input"* option was used and it is available through the W3C Validator site. The source code of each opened page in the Mozilla Firefox version 5.0 browser was copied to the validator code field. When the *"check"* is clicked on, the validator reported the problems found. After the analysis result, the validation system detected eight errors on the start page, four on the menu page, and three on the main use page. Some errors are not related to accessibility, and are related to not complying with W3C standards. The *Validator* does not cover all cases and, therefore, it was also necessary to open the source code of the evaluated system on a true text editor.

After the automatic validation and the code inspection, a scan of the system was executed using the NVDA screen reader. The reader helped to detect accessibility problems that could be corrected in the next step of the research. Many problems were detected since the study system was initially developed with no consideration about accessibility. The table I below describes the most common problems found:

TABLE I. RICH COMPONENTS AND PROBLEMS FOUND.

Component	Problem
rich:column	Cells of the table inaccessible via the keyboard. The screen reader does not read its contents.
rich:dataTable	
rich:message	Message generated by AJAX technology is not read by screen reader.
rich:modalPanel	Component inaccessible via the keyboard. The screen reader does not read its contents.
rich:panel	Component inaccessible via the keyboard.

Besides problems found in the rich components, problems in the use of not rich components were also identified, as listed below:

- Use of classic HTML tables for layout;
- Tab order not defined;
- Reading all content on the web pages without pause;
- Reading all the cells of the tables without pause;
- Table inaccessible via the keyboard;
- List items inaccessible via the keyboard;
- Selectable Radio buttons, but with limited access, since it is impossible to read the contents of rich:dataTable cells and to obtain information on the subject.

V. ADJUSTMENT OF THE SYSTEM ACCESSIBILITY

Based on the accessibility errors indicated by *Validator* and on the problems identified by the screen reader, accessibility adjustment was executed. Initially, the system was adapted to the basic accessibility recommendations WCAG 1 and WCAG 2, such as the deletion of tables used as layout and the correction of tags, both from the HTML part. Then, the WAI-ARIA attributes were inserted in the four pages, inside the HTML tags. It was necessary to change the *Document Type Definition* (DTD), a file that defines the valid

HTML blocks, to a version that recognizes those attributes and is specified by the W3C. *Role*, *aria-describedby* and *aria-live* attributes were inserted inside the HTML tags. *Jboss Richfaces* components do not recognize these attributes, because they were not built with WAI-ARIA support. Therefore, in order to use them, it was necessary to change the source code and compile a new version of the Jboss Richfaces toolkit, in which the *aria-describedby*, *role* and *tabindex* were inserted in seven of the components used in the system. *Jboss Richfaces* source code was changed in two ways: through a project manager software based on the *Project Object Model* (POM) concept, with the modification of files (usually with *.xml and *.jspx extensions) in which new attributes and insertion tags were inserted, respectively; and manually by opening the source code of each class of a given component and modifying/creating whatever was necessary.

TABLE II. RICH COMPONENTS, PROBLEMS FOUND AND SOLUTIONS

Component	Problem	Solution
rich:column	Cells of the table inaccessible via the keyboard	Putting "tabindex" attribute in each cell of the table to make them accessible.
rich:dataTable	Cells of the inaccessible via the keyboard	
rich:message	Message generated by AJAX technology is not read by screen reader	Putting HTML attribute "aria-live=assertive" inside div element responsible for displaying rich:message.
rich:modalPanel	Component inaccessible via the keyboard	This component is shown to user when he clicks to move from one page to another in order to inform him that the system is loading the next page. It was not modified, yet.
rich:panel	Component inaccessible via the keyboard	This component is used as layout organizer, so, it is not necessary to receive keyboard focus (in the studied system). But, Jboss Richfaces library was modified to recognize the "tabindex" attribute.

The dataTable rich component (table with AJAX support) was manually modified at the class in Java language that creates its cells (*cellRenderer.java*) to add the *role* and *tabindex* attributes, with values *"gridcell"* and *"-1"*, respectively. The first pair (*role="gridcell"*) indicates the role (table cell) and the second one (*tabindex="-1"*) makes it capable of receiving focus. On the other hand, an alternative to the need to modify the source code of this library and to compile a new version is to insert the attributes through the *JavaScript* language. In the same JRF dataTable component, this language was used to insert the *aria-describedby* attribute in each table cell, with the designation of the column header it belongs to. In addition, the source code developed in *JavaScript* was created to make possible the navigation on the tables using the direction keys 'left', 'up', 'right' and 'down', as well as to access the pages evaluation links and the radio button selection items using the 'enter' and 'space' keys. In a test using other browsers (Google Chrome, Opera, Internet Explorer) it was noticed that the source code has different behavior. Adjustments are necessary for the same operation in all possible browsers/screen readers. From the use of a screen reader to open the study system, it was found that words are spoken by a speaker with English language accent, which hinders the reading understanding of the site elements for users that are not proficient in this language. However, it is possible to use an external speaker, built in Brazilian Portuguese language, which improved the understanding of the spoken words by this research test subjects.

VI. STUDY VALIDATION

The system was tested by subjects with some visual disability. To increase the number of tests, subjects without disabilities also tested the system blindfold. All tests were executed using the NVDA version 2011.2 screen reader (a free software) and the Mozilla Firefox version 5.0 browser. It was noticed at this stage that because the screen reader was built before the arrival of RIAs, its use was difficult and it could not identify all the rich components. However, the NVDA has been constantly updated in order to support the WAI-ARIA tags, which may become accessible components.

VII. CONCLUSIONS AND FUTURE PROJECTS

If on one side, the technologies to develop rich interfaces for web sites advance, on the other side, the frequent use of these technologies contributes to decrease the accessibility of *Websites*. Faced with this problem, this paper performed a study to evaluate the accessibility of Web rich interface components. The paper analyses and corrects accessibility problems found in a system that uses the *Jboss Richface*s components library. The study demonstrates how it was possible to make the rich components of the *Jboss Richfaces 3.3.3.Final toolkit* accessible using only WAI-ARIA tags. To insert these tags, it is necessary to modify the *toolkit* source code (and to compile a new one). But, these tags could also be inserted using JavaScript language. Since the WAI-ARIA recommendations are generic, that is, they are not applicable to only a group of specific rich components; it is necessary to test these recommendations on other toolkits rich components (like Google Web Toolkit), in addition to execute tests with different browsers/screen readers. In summary, the study corroborates that it is possible to combine sites with rich interfaces and accessibility. However, currently this combination depends on the developer, limiting the dissemination of accessibility on the Web. This paper suggestion is for rich components to be made available to the developer in accordance with WAI-ARIA recommendations. This study presents a methodology to execute this task and it is the main contribution of the paper.

VIII. REFERENCES

[1] Brazil. Brazilian Association of Technical Standards (ABNT). Brazilian Standard ABNT NBR 9050: Accessibility of buildings, furniture, equipment and urban spaces. 2nd Edition, 2004. Available at: http://www.mpdft.gov.br/sicorde/NBR9050-31052004.pdf. Accessed on: 29 Nov 2011.

[2] P. Deitel and H. Deitel. Ajax, rich internet applications and web development for programmers. Ed. Pearson. 2008.

[3] Web Accessibility Initiative – Accessible Rich Internet Applications Suite (WAI-ARIA). Available at: http://www.w3.org/WAI/intro/aria.php. Accessed on: 29 Nov 2011.

[4] Cynthia Says. Available at: http://www.cynthiasays.com/ Accessed on: 29 Nov 2011.

[5] Hera. Available on: http://www.sidar.org/hera/index.php.en Accessed on: 29 Nov 2011

[6] Ocawa. Available at: http://www.ocawa.com/en/Test-your-Web-Site.htm Accessed on: 29 Nov 2011.

[7] W3C. Markup Validation Service. Available at: http://validator.w3.org/. Accessed on: 29 Nov 2011.

[8] L. H. Tonet and D. P. Andrés. Research of tools of computer accessibility for visually impaired and W3C Recommendations. 2006. Available at: http://guaiba.ulbra.tche.br/pesquisa/2006/artigos/sistemas/161.pdf Accessed on: 29 Nov 2011.

[9] NonVisual Desktop Acess (NVDA). Available at: http://www.nvda-project.org/. Accessed on: 29 Nov 2011.

[10] Jaws for Windows Screen Reading. Available at: http://www.freedomscientific.com/products/fs/jaws-product-page.asp Access on: 29 Nov 2011.

[11] Virtual Vision. Available at: http://www.virtualvision.com.br/index.html Accessed on: 29 Nov 2011.

[12] Window-Eyes. Available at: http://www.gwmicro.com/window-eyes/ Accessed on: 29 Nov 2011.

[13] Brazil Accessibility. DaSilva Evaluator. Available at: http://www.dasilva.org.br/?blogid=2. Accessed on: 29 Nov 2011.

[14] Brazil. Ministry of Planning, Budget and Management. Evaluator and Simulator Accessibility of Sites. Available at: http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG/ases-avaliador-e-simulador-de-acessibilidade-sitios. Accessed on: 29 Nov 2011.